Results **1 - 10** of about **1,840,000** for **c programing library "keywords"**. (**0.08** seconds)

Book results for **c programing library "keywords"**

C++ Primer Plus - by Stephen Prata - 1098 pages
C - by Herbert Schildt - 805 pages
C++ Programming for the Absolute Beginner - by Dirk Henkemans, Mark Lee - 481 pages

#665 (load **library**("**c**:**program** files\common files\ole db\comsvcs ...

Regsvr32 "**C**:**Program** Files\Common Files\System\Ole DB\comsvcs.dll". but it is giving an error as. load **library**("**c**:**program** files\common files\ole ...
trac.edgewall.org/ticket/665 - 27k - Cached - Similar pages

Managed Extensions for **C++ Programming** (C++)

Managed Extensions for **C++** Windows Applications: Provides links to topics describing how to link your Windows applications to the **C** run-time **library** and how ...
msdn2.microsoft.com/en-us/**library**/aa712574(vs.71).aspx - 19k - Cached - Similar pages

New Language Features in Visual **C++**

Some specifiers are also valid for native **programming**. For more information, see How to: Declare ... The following **keywords** have been added to Visual **C++**. ...
msdn2.microsoft.com/en-us/**library**/xey702bw(VS.80).aspx - 32k - Cached - Similar pages

C++ Programming/Keywords - Wikibooks, collection of open-content ...

C++ Programming/Keywords ... < **C++ Programming**. Jump to: navigation, search ... of reserved identifiers carried over from the **C library** specification. ...
en.wikibooks.org/wiki/**C++**_Programming/**Keywords** - 18k - Cached - Similar pages

C (**programming** language) - Wikipedia, the free encyclopedia

C program source text is free-format, using semicolon as a statement terminator (not a delimiter). **C** has around 30 reserved **keywords**. ...
en.wikipedia.org/wiki/**C**_(programming_language) - 111k - Cached - Similar pages

keywords" content="create source code, edit prepared **C** source code ...

C library supplies pre-compiled routines to enable **program** to run. If an error occurs either there is an error in the **program** or the routine cannot be found ...
www.geocities.com/learnprogramming123/**C**lesson1.htm - 26k - Cached - Similar pages

Amazon.com: Advanced **C++ Programming** Styles and Idioms: Books ...

C++ Programming Style (Addison-Wesley Professional Computing Series) by Tom ..... good book <<The **C++** Standard **Library** : A Tutorial and Reference>> as well. ...
www.amazon.com/Advanced-**C**-Programming-Styles-Idioms/dp/0201548550 - 199k - Cached - Similar pages

Amazon.com: **C Programming** Language, 2nd Ed: Books: Brian W ...

The **C++ Programming** Language (Special 3rd Edition) by Bjarne Stroustrup. (272) $56.59 ... **Programming** in **C** (3rd Edition) (Developer's **Library**) ...
www.amazon.com/**C**-Programming-Language-2nd-Ed/dp/0131103709 - 181k - Cached - Similar pages

Day of 1/24/2002: Top 10 Search **Keywords** by Server Used

1: gd **library**; MSN: 11: mapedit 10: 10: baklava 7: mailto 5: shared calendar 5: hummus

recipe 5: hummus 4: **c programing** 4: usenet history **...**
www.boutell.com/wusage/example/daily/2002/01/24/search**keywords**byserver.html - 6k - Cached - Similar pages

Herb Sutter's Blog : **C++**/CLI **keywords**: Under the hood
All the other **keywords**, below, are contextual **keywords** that do not conflict with identifiers. Any legal ISO **C++ program** that already uses the names below as **...**
blogs.msdn.com/hsutter/archive/2003/11/23/53519.aspx - 36k - Cached - Similar pages

1 2 3 4 5 6 7 8 9 10 **Next**

C++ Programming/Keywords

From Wikibooks, the open-content textbooks collection

ISO C++ (C++98) Keywords

- `and`
- `and_eq`
- `asm`
- `auto`
- `bitand`
- `bitor`
- `bool`
- `break`
- `case`
- `catch`
- `char`
- `class`
- `compl`
- `const`
- `const_cast`
- `continue`
- `default`
- `delete`
- `do`
- `double`
- `dynamic_cast`
- `else`
- `enum`
- `explicit`
- `export`
- `extern`
- `false`
- `float`
- `for`
- `friend`
- `goto`
- `if`
- `inline`
- `int`
- `long`
- `mutable`
- `namespace`
- `new`
- `not`
- `not_eq`
- `operator`
- `or`
- `or_eq`
- `private`
- `protected`
- `public`
- `register`
- `reinterpret_cast`
- `return`
- `short`
- `signed`
- `sizeof`
- `static`
- `static_cast`
- `struct`
- `switch`
- `template`
- `this`
- `throw`
- `true`
- `try`
- `typedef`
- `typeid`
- `typename`
- `union`
- `unsigned`
- `using`
- `virtual`
- `void`
- `volatile`
- `wchar_t`
- `while`
- `xor`
- `xor_eq`

Specific compilers may (in a non-standard compliant mode) also treat some other words as keywords, including `cdecl`, `far`, `fortran`, `huge`, `interrupt`, `near`, `pascal`, `typeof`. Old compilers may recognize the `overload` keyword, an anachronism that has been removed from the language.

The next revision of C++, informally known as C++0x for now, is likely to add some keywords, probably including at least:

- `static_assert`
- `decltype`
- `nullptr`

(These are being considered carefully to minimize breakage to existing code; see http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2105.html for some details.)

Old compilers may not recognize some or all of the following keywords:

- `and`
- `and_eq`
- `bitand`
- `bitor`
- `bool`
- `catch`
- `dynamic_cast`
- `explicit`
- `export`
- `false`
- `mutable`
- `namespace`
- `or`
- `or_eq`
- `reinterpret_cast`
- `static_cast`
- `template`
- `throw`
- `typeid`
- `typename`
- `using`
- `wchar_t`
- `xor`
- `xor_eq`

- `compl`
- `const_cast`
- `not`
- `not_eq`
- `true`
- `try`

C++ Reserved Identifiers

Some "nonstandard" identifiers are reserved for distinct uses, to avoid conflicts on the naming of identifiers by vendors, library creators and users in general.

Reserved identifiers include keywords with two consecutive underscores (__), all that start with an underscore followed by an uppercase letter, and some other categories of reserved identifiers carried over from the C library specification.

TODO
It would be nice to list those C reserved identifiers

2.4. Keywords and identifiers

After covering the underlying alphabet, we can look at more interesting elements of C. The most obvious of the language elements are *keywords* and *identifiers*; their forms are identical (although their meanings are different).

2.4.1. Keywords

C keeps a small set of *keywords* for its own use. These keywords cannot be used as identifiers in the program — a common restriction with modern languages. Where users of Old C may be surprised is in the introduction of some new keywords; if those names were used as identifiers in previous programs, then the programs will have to be changed. It will be easy to spot, because it will provoke your compiler into telling you about invalid names for things. Here is the list of keywords used in Standard C; you will notice that none of them use upper-case letters.

`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

Table 2.3. Keywords

The new keywords that are likely to surprise old programmers are: `const`, `signed`, `void` and `volatile` (although `void` has been around for a while). Eagle eyed readers may have noticed that some implementations of C used to use the keywords `entry`, `asm`, and `fortran`. These are not part of the Standard, and few will mourn them.

2.4.2. Identifiers

Identifier is the fancy term used to mean 'name'. In C, identifiers are used to refer to a number of things: we've already seen them used to name variables and functions. They are also used to give names to some things we haven't seen yet, amongst which are *labels* and the 'tags' of *structures*, *unions*, and *enums*.

The rules for the construction of identifiers are simple: you may use the 52 upper and lower case alphabetic characters, the 10 digits and finally the underscore '_', which is considered to be an alphabetic character for this purpose. The only restriction is the usual one; identifiers *must* start with an alphabetic character.

Although there is no restriction on the length of identifiers in the Standard, this is a point that needs a bit of explanation. In Old C, as in Standard C, there has *never* been any restriction on the length of identifiers. The problem is that there was never any guarantee that more than a certain number of characters would be checked when names were compared for equality—in Old C this was eight characters, in Standard C this has changed to 31.

So, practically speaking, the new limit is 31 characters—although identifiers *may* be longer, they must differ in the first 31 characters if you want to be sure that your programs are portable. The Standard allows for implementations to support longer names if they wish to, so if you do use longer names, make sure that you don't rely on the checking stopping at 31.

One of the most controversial parts of the Standard is the length of *external identifiers*. External identifiers are the ones that have to be visible outside the current source code file. Typical examples of these would be library routines or functions which have to be called from several different source files.

The Standard chose to stay with the old restrictions on these external names: they are not guaranteed to be different unless they differ from each other in the first six characters. Worse than that, upper and lower case letters may be treated the same!

The reason for this is a pragmatic one: the way that most C compilation systems work is to use operating

system specific tools to bind library functions into a C program. These tools are outside the control of the C compiler writer, so the Standard has to impose realistic limits that are likely to be possible to meet. There is nothing to prevent any specific implementation from giving better limits than these, but for maximum portability the six monocase characters must be all that you expect. The Standard warns that it views both the use of only one case and any restriction on the length of external names to less than 31 characters as obsolescent features. A later standard may insist that the restrictions are lifted; let's hope that it is soon.

Previous section | Chapter contents |
Next section

C++ Programming/Keywords

From Wikibooks, the open-content textbooks collection

ISO C++ (C++98) Keywords

- `and`
- `and_eq`
- `asm`
- `auto`
- `bitand`
- `bitor`
- `bool`
- `break`
- `case`
- `catch`
- `char`
- `class`
- `compl`
- `const`
- `const_cast`
- `continue`
- `default`
- `delete`
- `do`
- `double`
- `dynamic_cast`
- `else`
- `enum`
- `explicit`
- `export`
- `extern`
- `false`
- `float`
- `for`
- `friend`
- `goto`
- `if`
- `inline`
- `int`
- `long`
- `mutable`
- `namespace`
- `new`
- `not`
- `not_eq`
- `operator`
- `or`
- `or_eq`
- `private`
- `protected`
- `public`
- `register`
- `reinterpret_cast`
- `return`
- `short`
- `signed`
- `sizeof`
- `static`
- `static_cast`
- `struct`
- `switch`
- `template`
- `this`
- `throw`
- `true`
- `try`
- `typedef`
- `typeid`
- `typename`
- `union`
- `unsigned`
- `using`
- `virtual`
- `void`
- `volatile`
- `wchar_t`
- `while`
- `xor`
- `xor_eq`

Specific compilers may (in a non-standard compliant mode) also treat some other words as keywords, including `cdecl`, `far`, `fortran`, `huge`, `interrupt`, `near`, `pascal`, `typeof`. Old compilers may recognize the `overload` keyword, an anachronism that has been removed from the language.

The next revision of C++, informally known as C++0x for now, is likely to add some keywords, probably including at least:

- `static_assert`
- `decltype`
- `nullptr`

(These are being considered carefully to minimize breakage to existing code; see http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2105.html for some details.)

Old compilers may not recognize some or all of the following keywords:

- `and`
- `and_eq`
- `bitand`
- `bitor`
- `bool`
- `catch`
- `dynamic_cast`
- `explicit`
- `export`
- `false`
- `mutable`
- `namespace`
- `or`
- `or_eq`
- `reinterpret_cast`
- `static_cast`
- `template`
- `throw`
- `typeid`
- `typename`
- `using`
- `wchar_t`
- `xor`
- `xor_eq`

- `compl`
- `const_cast`
- `not`
- `not_eq`
- `true`
- `try`

C++ Reserved Identifiers

Some "nonstandard" identifiers are reserved for distinct uses, to avoid conflicts on the naming of identifiers by vendors, library creators and users in general.

Reserved identifiers include keywords with two consecutive underscores (__), all that start with an underscore followed by an uppercase letter, and some other categories of reserved identifiers carried over from the C library specification.

TODO
It would be nice to list those C reserved identifiers